ZENITH OAR: Week Two

The Tech Club - Robotics Team

Now before simulating more complex tasks we will need to know some basics of control theory in order to work properly with robots.

The relevance of control theory to robotics is explained in the following example using a simple robot. Let us consider an example of building a robot that can travel at a specific velocity. Now, you can simply give power to the motors and hope that they're producing the speed we desire.

This is not the case when you'd want precision, which is often necessary in robotics. You'd want to measure the speed of the robot and compensate for any difference(error) in speed you may see. Now, these differences may arise due to multitude of reasons due to uncertainties. It is also applicable to these systems as well.

Thus, having a feedback mechanism allows us to correct the system and move it to the desired speed. This is essentially what control theory is all about. It's even used in rocket systems, flight controls and much more. So let's learn a little bit about control theory, shall we?

Basics

- State = Representation of what the system is currently doing
- Dynamics = Description of how the state changes
- Reference = What we want the system to do
- Output = Measurement of (some aspects of the) system
- Input = Control signal
- Feedback = Mapping from outputs to inputs

Controllers

A block diagram is a pictorial representation of the cause and effect relationship between the input and output of a physical system.

A block diagram provides a means to easily identify the functional relationships among the various components of a control system.

The figure below shows a basic feedback control system as represented by a block diagram. The functional relationships between these elements are easily seen.

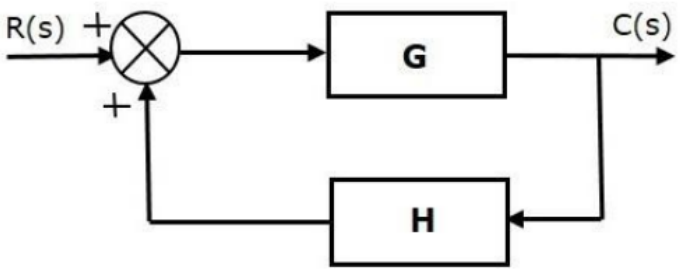

Figure 1: Basic Feedback control system (closed loop)

The block G is the plant or the system or the process through which a particular quantity is controlled. The block H is the feedback path gain, since it goes from output to the input. R(s) is the input and C(s) is the output.

Now we can also represent this same block diagram using a single equation called a transfer function. Consider a simple system which takes a voltage input and gives voltage output. Now we all know that this is a transformer. Now we can represent this system simply by a transfer function which will give us the output of the system for any input. If the input is say X(s), then the output is given by X(s) * tf(s), where tf(s) is the transfer function of the system. The transfer function of the system given in Figure 1 is,

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 - G(s) \cdot H(s)}$$

A system is said to be a control system when we know the required output and are trying to drive the system to that desired value. This can be achieved using a controller.

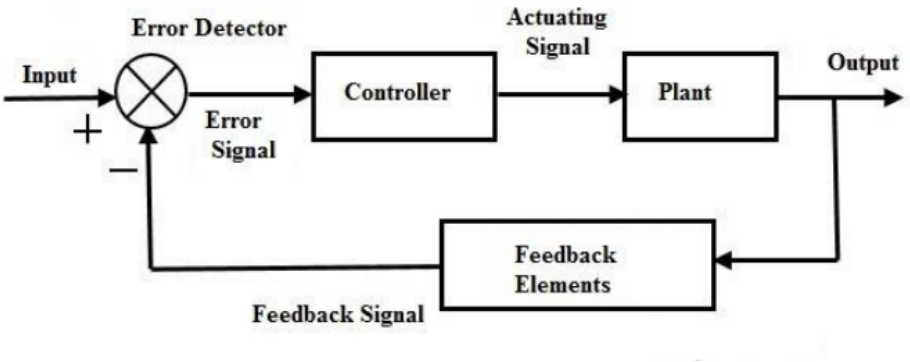

Figure 2: Typical controller block diagram

The difference between the input (the desired value) and the observed output is generally ERROR. This ERROR is passed as a signal through the controller and the plant, which are designed specifically to reduce the error in the next loop.

- For example, consider the temperature control system inside our ACs. When we set the temperature to, say 20* C, we are providing this signal as the input, or the set point (SP) of this system is 20* C.
- The controller looks at the set point and compares it with the actual value of the room temperature. This actual value is also called the Process Value (PV), as returned by the feedback element.
- If the SP and the PV are the same – then the controller need not do anything. It will set its output to zero.
- However, if there is a difference between the SP and the PV we have an error and a correction is required. In our house, this will either be cooling or heating depending on whether the PV is higher or lower than the SP.

So, the error becomes zero when the current output value is same as that of the desired output and until then, the controllers and the feedback elements will try to reduce the error.

Types of controllers

The controller that we shall we covering is the PID (Proportional Integral Differential) Controller. As we saw in the temperature example, the controller takes the PV and SP signal, which is then sent through a block to calculate controller output. This output is sent to an actuator which controls the movement.

We are interested here in what the block actually does. Specifically, we are interested in the calculations in the block, made using the SP and PV signals. These calculations, called the "Modes of Control" include the following operations.

- Proportional (P)
- Integral (I)
- Derivative (D)

Here is how a PID Controller looks on the inside:

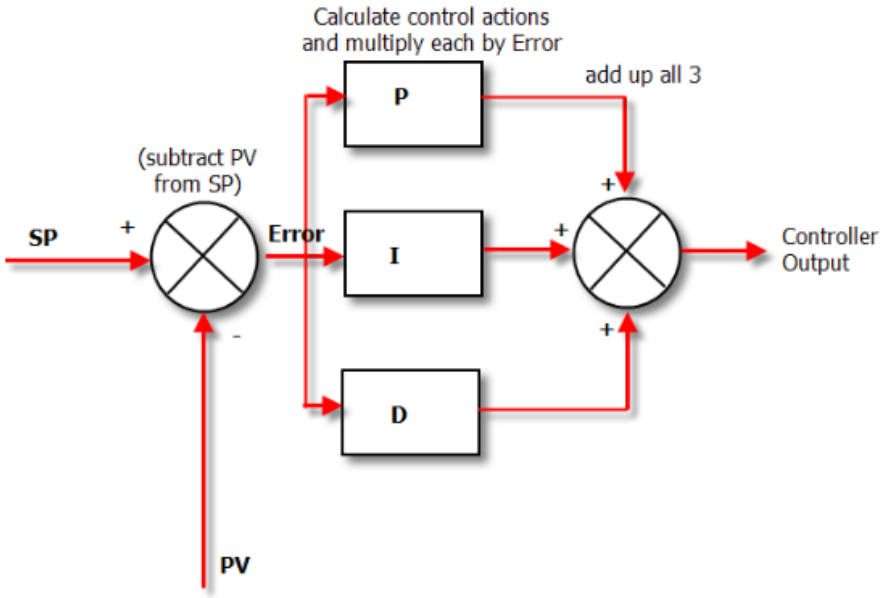

Figure 3: PID Controller

The PV signal is subtracted from the SP signal to return the ERROR signal. The error is simply multiplied by one, two or all of the calculated P, I and D actions (depending which ones are switched on). Then the resulting values are added together and sent to the controller output. These 3 modes are used in different combinations, shown below.

- P – Sometimes used
- PI - Often used
- PID – Sometimes used
- PD – as rare as SSN semester holidays but can be useful for controlling servomotors.

If y(t) is the controller output, then the final form of a generic PID controller is given by:

$$y(t) = K_p \times e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$

where,
K_p is the proportional gain, a tuning parameter,
K_i is the integral gain, a tuning parameter,
K_d is the derivative gain, a tuning parameter,
e(t) = SP - PV is the error (SP is the setpoint, and PV is the process variable)

Refer to Useful Link 4 for demo of PID control.

Proportional

The proportional term produces an output value that is proportional to the current error value. A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances.

Integral

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral term in a PID controller gives the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (K_i) and added to the controller output. The integral term accelerates the movement of the process towards set point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set point value.

Differential

Derivative action predicts system behaviour and thus improves settling time and stability of the system. Derivative action is seldom used in practice because of its variable impact on system stability in real-world applications. A high K_d can decrease the overshoot resulting from the integral term, but the big problem with D control is that if you have noise on your signal this confuses the hell out of the algorithm. And your control output jumps all over the place, messing up your control. So unless PI control is really slow, don't worry about switching D on.

Assignment

For this week's assignment, you will have to implement a PID controller for cruise control. You will find the base matlab code in the GitHub repository. You will be given the system defining the vehicle parameters. You will have to write a transfer function in s-domain defining the PID controller (the vehicle system is also modelled in s-domain, refer to that if you have any doubts). The code will also contain the plotting part which will plot the response of your controller with the vehicle.Design a controller such that the velocity

1) Never reaches the reference value
2) Overshoots above the set reference value but returns to the reference value.
3) Reaches the reference value steadily without any overshoot.

Hint: Vary the parameters Kp, Ki, Kd.

You will have to test different values to achieve the given controller requirements and submit a report containing your inferences on the role each parameter plays in the controller.

Additionally if you are interested, try to simulate PID cruise control for the pioneer robot as well. You will have to give a certain reference velocity as input and you will have to send control signals to the unicycle model you built last week (just set the linear velocity, cruise control is just about going straight) by calculating the input using P, PI, and PID controllers to achieve that speed. You will be using this idea in the next week's assignment so give it a try.

Hint: To find the error, you will have to find the actual velocity of the robot. Lookup the remote API of Coppeliasim to find out how to do this.

Useful Links and References

1. Let's talk about PID controllers: What are they? And why do we need them? | by Hamza | Medium - Article
2. Introduction to Robotic control systems - Article
3. PID Control - A Brief introduction | Brian Douglas - Video tutorial
4. What PIDs do and how they do it | RCModelReviews - Video tutorial
5. Remote API Documentation